$$\sum_{i=1}^{m} \sum_{q=\max\{0,j-w_i+1\}}^{\min\{L-w_i,j\}} \sum_{r=\max\{0,k-h_i+1\}}^{\min\{L-h_i,k\}} x_{iqr} \leq 1, \quad j,k = 0,\ldots,L-1 \tag{161}$$

e vengono usati al posto di (159)

Esempio 17 Si consideri il caso $m = 5$, $L = 10$, $w = h = (7,5,4,4,2)$ (ovverosia gli oggetti sono tutti quadrati) e si scriva il vincolo (161) associato a $j = 3$ e $k = 4$

Il Caso più Frequente in Pratica

Nella maggior parte dei casi pratici il problema da risolvere è un problema di Bin Packing Bidimensionale

Anzichè formularlo tramite la variante menzionata del modello sopra, il problema viene formulato tramite un modello del tipo (37), in cui $\mathcal{S}$ rappresenta la collezione dei sottoinsiemi massimali di oggetti (rettangoli) che possono essere impaccati in un bin (quadrato), ed il modello illustrato per il Knapsack Bidimensionale risolve il corrispondente problema di generazione di colonne (con profitti degli oggetti dati dalle variabili duali associate)

Scheduling

I problemi di *scheduling* si riferiscono all'organizzazione temporale del lavoro che deve essere svolto in un generico processo produttivo

Data l'ampia gamma di situazioni reali possibili, il numero di problemi di scheduling di interesse pratico è molto vasto, e non è possibile individuare un problema "base" di riferimento

Per consentire l'identificazione dei problemi all'interno di una classe così vasta, è stata introdotta addirittura una sintassi apposita che specifica le caratteristiche fondamentali di ciascun problema (tale sintassi non verrà illustrata qui)

In linea di massima, in un problema di scheduling sono dati n *lavori* che devono essere eseguiti da m *macchine*, e si deve decidere l'assegnazione dei lavori alle macchine e/o l'ordine in cui ciascuna macchina esegue i lavori assegnatile

L'istante a partire da cui le macchine possono lavorare è convenzionalmente fissato a 0, e ciascuna macchina può eseguire un solo lavoro alla volta

Ciascun lavoro j è caratterizzato da:

- un insieme di *operazioni* in cui è suddiviso (eventualmente *una sola*, nel qual caso il lavoro e la sua operazione sono la stessa cosa)

- il tempo di esecuzione (*processing time*) di ciascuna operazione su ciascuna macchina (eventualmente *infinito* se la macchina non può eseguire l'operazione)
- un istante r_j prima del quale non può essere iniziato (*release date*) (eventualmente 0)
- un istante d_j entro il quale dovrebbe essere terminato (*due date*) (eventualmente ∞)

Le macchine possono essere:

- *identiche*, ovverosia qualunque operazione può essere eseguita da ciascuna macchina nello stesso tempo
- *uniformi*, ovverosia ciascuna ha una sua *velocità* e qualunque operazione può essere eseguita da ciascuna macchina in un tempo inversamente proporzionale alla velocità della macchina
- *differenti*, ovverosia il tempo di esecuzione di una operazione cambia da macchina a macchina (e spesso l'operazione può essere eseguita solo da alcune macchine)

Caratteristiche aggiuntive dei problemi possono essere:

- la possibilità di *interrompere* l'esecuzione di un'operazione riprendendola successivamente (eventualmente su una machina diversa), detta *preemption*
- l'esistenza di *vincoli di precedenza* tra operazioni di uno stesso lavoro e di lavori diversi

Gli obiettivi sono generalmente dati dalla *minimizzazione* della somma di o del massimo tra:

- *istante di completamento* C_j del lavoro j
- *lateness* $L_j := C_j - d_j$ del lavoro j (penalizzando il ritardo rispetto alla due date e premiando l'eventuale anticipo)
- *tardiness* $L_j := \max\{0, C_j - d_j\}$ del lavoro j (penalizzando il ritardo rispetto alla due date ma non premiando l'eventuale anticipo)

L'obiettivo di gran lunga più frequente è la minimizzazione del massimo istante di completamento di un lavoro, detto *makespan* $C_{\max} := \max_{j=1}^{n} C_j$ – in altre parole si vogliono terminare tutti i lavori il prima possibile

Nel seguito verranno illustrati due esempi rilevanti, il secondo dei quali fonirà lo spunto per discutere il fatto che, per molti problemi di scheduling di interesse pratico, non si conoscono modelli ILP "forti"

L'effetto del non conoscere modelli ILP "forti" ha come conseguenza che, per molti problemi di interesse pratico, non solo non si conosce il valore della soluzione ottima, ma non si conoscono neppure lower bound "buoni" su di essa che possano certificare la qualità delle soluzioni euristiche determinate

Macchine Identiche Parallele, o $P||C_{\max}$

Il problema delle *Macchine Identiche Parallele*, indicato con $P||C_{\max}$ secondo la sintassi menzionata sopra, corrisponde al caso in cui ciascun lavoro j ha una sola operazione p_j con tempo di esecuzione p_j su una qualunque macchina (oltre ad avere $r_j = 0$ e $d_j = \infty$), e si vogliono assegnare i lavori alle macchine in modo da minimizzare il makespan

Esempio 18 Si consideri il caso $m = 2$, $n = 4$, $p = (3, 1, 4, 3)$ e si determini la soluzione ottima

Per definire un modello ILP per il problema, si osservi innanzitutto che l'unica decisione da prendere corrisponde all'assegnazione dei lavori alle macchine e *non* all'ordine in cui le macchine eseguono i lavori assegnati loro, in quanto il makespan non dipende da quest'ultimo, essendo dato dalla massima somma dei tempi di esecuzione dei lavori assegnati ad una macchina

Un'interpretazione possibile del problema visto è la variante del Bin Packing in cui il numero m di bin utilizzati è fissato mentre si può variare la capcità b, per la quale si vuole trovare il valore *minimo* che garantisce che gli oggetti siano impaccabili negli m bin

Introducendo le variabili:

$$x_{ij} := \begin{cases} 1, & \text{se il lavoro } j \text{ è assegnato alla macchina } i \\ 0, & \text{altrimenti} \end{cases}$$

un modello con funzione obiettivo nonlineare è dato da:

$$\min \max_{i=1}^{m} \sum_{j=1}^{n} p_j x_{ij} \tag{162}$$

$$\sum_{i=1}^{m} x_{ij} = 1, \quad j = 1, \ldots, n \tag{163}$$

$$x_{ij} \in \{0, 1\}, \quad i = 1, \ldots, m, \ j = 1, \ldots, n \tag{164}$$

Come per molti altri problemi di scheduling (ed anche altri problemi di tipo "min max") un modello ILP è ottenuto introducendo la variabile z e sostituendo (162) con:

$$\min z \tag{165}$$

$$\sum_{j=1}^{n} p_j x_{ij} \quad \leq \quad z, \quad i = 1, \ldots, m \tag{166}$$

Il modello ILP visto ha tutti i difetti del modello ILP "debole" per il Bin Packing

A differenza del Bin Packing non esiste un modello "forte" in questo caso, e la cosa migliore da fare per risolvere il problema è determinare il valore ottimo di z provando diversi valori e verificando che siano ammissibili risolvendo il corrispondente modello di bin packing con capacità z tramite il modello "forte" (la sequenza di valori di z è guidata dalla cosiddetta *ricerca binaria*, non illustrata qui)

Flow Shop, o $F||C_{\max}$

Il problema del *Flow Shop*, indicato con $F||C_{\max}$ secondo la sintassi menzionata sopra, appartiene alla famiglia dei problemi di tipo "Shop" (che include il *Job Shop* e l'*Open Shop*) che, tra i problemi combinatori facili da formulare, sono tra i più difficili da risolvere

Nel Flow Shop, ciascun lavoro j è suddiviso in m operazioni, l'i-esima delle quali deve essere eseguita sulla macchina i, ha tempo di esecuzione p_{ij}, e può iniziare solo dopo che sia stata completata l'$i-1$-esima (la prima operazione puó iniziare all'istante 0, cioè $r_j = 0$; inoltre $d_j = \infty$)

Si vuole decidere, per ogni macchina, l'ordine di esecuzione delle operazioni dei vari lavori in modo da minimizzare il makespan

Esempio 19 Si consideri il caso $m = n = 3$, $p = \begin{pmatrix} 5 & 4 & 2 \\ 8 & 7 & 4 \\ 3 & 6 & 5 \end{pmatrix}$ e si determini una soluzione euristica

Introducendo le variabili continue:

t_{ij} := istante di inizio dell'i-esima operazione del lavoro j (sull'i-esima macchina)

il modello più naturale per il problema, con vincoli nonlineari, è dato da:

$$\min z \tag{167}$$

$$t_{mj} + p_{mj} \leq z, \quad j = 1, \ldots, n \tag{168}$$

$$t_{(i-1)j} + p_{(i-1)j} \leq t_{ij}, \quad j = 1, \ldots, n,\ i = 2, \ldots, m \tag{169}$$

$$t_{ij} + p_{ij} \leq t_{ik} \quad \textbf{or} \quad t_{ik} + p_{ik} \leq t_{ij}, \quad j, k = 1, \ldots, n,\ j \neq k,\ i = 1, \ldots, m \tag{170}$$

$$t_{1j} \geq 0, \quad j = 1, \ldots, n \tag{171}$$

dove i vincoli (169) impongono l'ordine di esecuzione delle operazioni di un lavoro ed i vincoli nonlineari (170) impongono che ciascuna macchina i esegua al più una operazione alla volta (l'i-esima operazione del lavoro j viene eseguita prima dell'i-esima operazione del lavoro k, *oppure* l'i-esima operazione del lavoro k viene eseguita prima dell'i-esima operazione del lavoro j)

Per linearizzare i vincoli (170) occorre, come nel caso delle finestre temporali per il problema del Vehicle Routing, introdurre variabili binarie che esprimano la precedenza tra le operazioni di una stessa macchina ed un coefficiente "big M":

$$x_{ijk} := \begin{cases} 1, & \text{se l'}i\text{-esima operazione del lavoro } j \text{ precede l'}i\text{-esima operazione del lavoro } k \\ 0, & \text{altrimenti} \end{cases}$$

sostituendo (170) con:

$$x_{ijk} + x_{ikj} = 1, \quad j, k = 1, \ldots, n,\ j < k,\ i = 1, \ldots, m \tag{172}$$

$$t_{ij} + p_{ij} \leq t_{ik} + M x_{ikj}, \quad j, k = 1, \ldots, n,\ j \neq k,\ i = 1, \ldots, m \tag{173}$$

$$x_{ijk} \in \{0, 1\}, \quad j, k = 1, \ldots, n,\ j \neq k,\ i = 1, \ldots, m \tag{174}$$

La presenza del "big M" rende estremamente "debole" il rilassamento continuo del modello visto, ma d'altra parte per il problema del Flow Shop (così come per gli altri problemi di tipo Shop e per molti altri problemi di Scheduling) non si conoscono modelli ILP "forti"